SIGMATEK

3,5“ Terminal

Projektrealisierung

Erstellungsdatum: 10.02.2014 | Versionsdatum: 28.04.2016 | Artikelnummer: XX-XXX-XXXX-D

Herausgeber: SIGMATEK GmbH & Co KG
A-5112 Lamprechtshausen
Tel.: 06274/4321
Fax: 06274/4321-18
Email: office@sigmatek.at
WWW.SIGMATEK-AUTOMATION.COM

Copyright © 2014
SIGMATEK GmbH & Co KG

Originalsprache

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder in einem anderen Verfahren) ohne ausdrückliche Genehmigung reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Inhaltliche Änderungen behalten wir uns ohne Ankündigung vor. Die SIGMATEK GmbH & Co KG haftet nicht für technische oder drucktechnische Fehler in diesem Handbuch und übernimmt keine Haftung für Schäden, die auf die Nutzung dieses Handbuches zurückzuführen sind.

Projektrealisierung 3,5“ Terminal

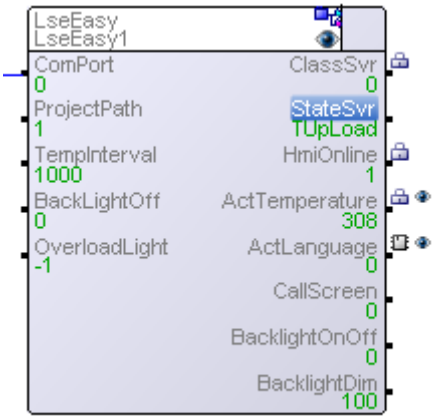
Diese Dokumentation dient als Leitfaden zur Projektrealisierung auf einem 3,5“ Terminal.

Inhaltsverzeichnis

1 Grundsätzliche Funktionsweise

Das Display kommuniziert mittels CAN-Bus mit der CPU. Es wird eine Visualisierung mit dem LASAL Screen Editor erstellt, welche auf die CPU übertragen wird (Files auf der CPU).

Die Klasse LSEEasy baut eine Kommunikation zum Display auf. Das eingestellte Visualisierungsprojekt wird geladen. Die Klasse prüft, ob das Projekt im Display gleich dem Projekt ist, welches auf der Steuerung ist. Besteht ein Unterschied, wird das Projekt von der Klasse automatisch zum Display geschickt und dort gespeichert. Dies kann man am StateSvr der Klasse LSEEasy beobachten.



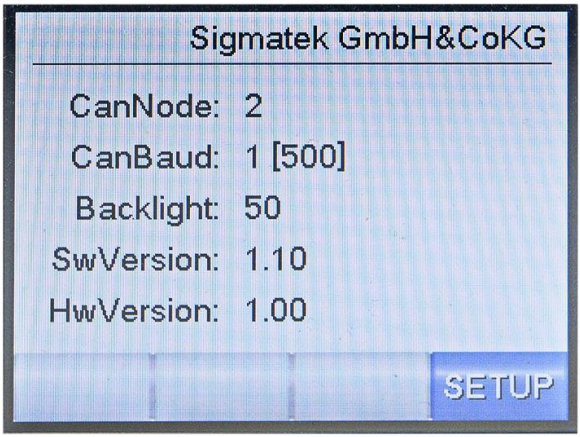
Das Display zeigt während des Downloads folgendes an: Im UPLOAD PROJECT Balken wird der Fortschritt des Downloads angezeigt. Im UNDER CONSTRUCTION Balken läuft ständig ein Balken durch. Dieser kann jedoch ab und zu an der gleichen Stelle verweilen, da dann das Programm in den Speicher des Displays geschrieben wird.

Anschließend wird die Visualisierung gestartet. Ist das Projekt gleich, wird die Visualisierung sofort gestartet.

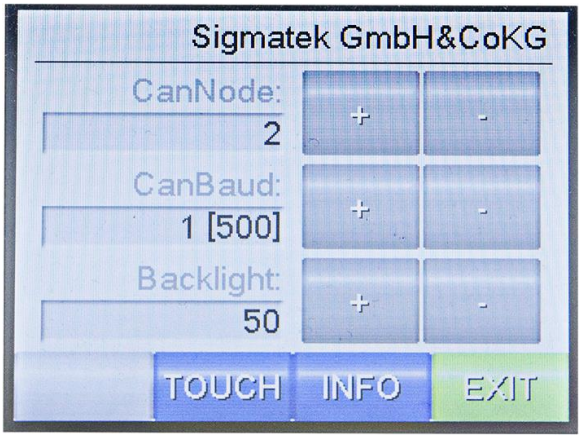
2 Einstellungen auf der CPU-Seite

```
autoexec.lsl :SET CAN 1 BAUD 1 STATION 0
```

3 Einstellungen auf dem 3,5 Zoll Display



SETUP Button drücken!



CanNode : einstellen.

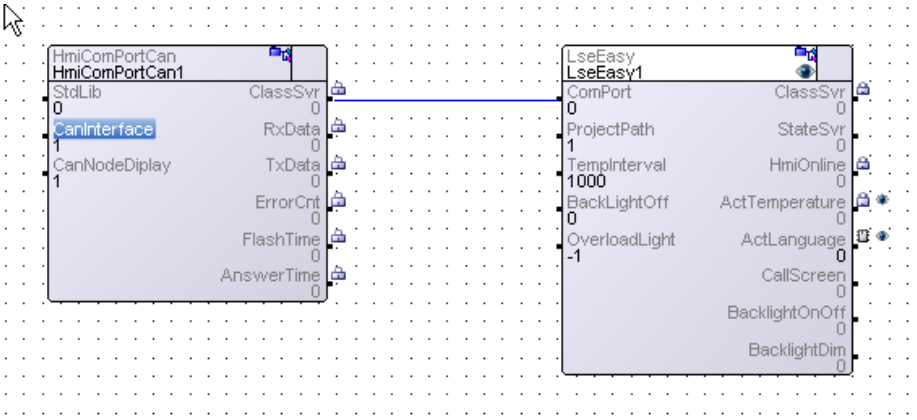
CanBaud : einstellen

Touch: Dient als Touch-Test

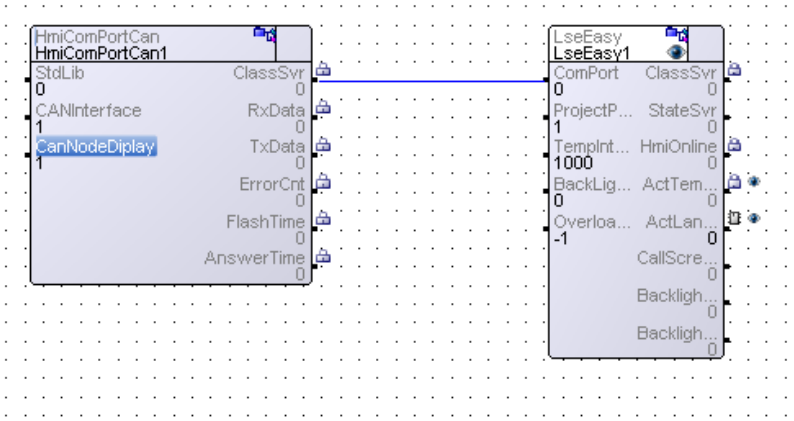
RET: Rückkehr zum Hauptmenü

4 Erstellen eines Class Projekts zur Ansteuerung eines 3,5 Zoll Displays

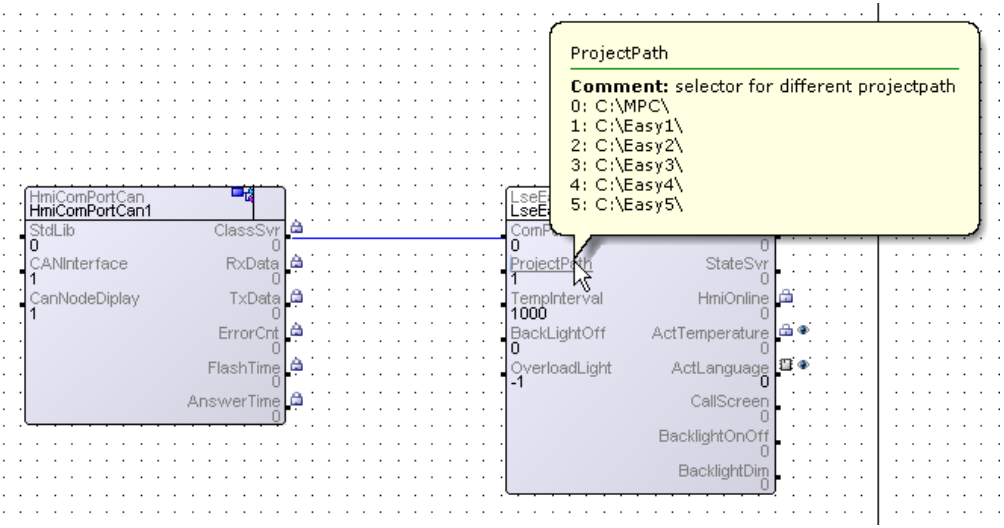
- Legen Sie ein Netzwerk an und platzieren Sie folgende Objekte darauf. Stellen Sie das CanInterface ein (1 .. Can1 2.. Can2).



- Stellen Sie das CanNodeDisplay ein (CanNode, welches am Display im Setup eingestellt wurde).



- Weiters ist beim Client ProjectPath ein Initwert anzugeben. Dieser ist davon abhängig, von welchem Pfad die Visualisierung für das Display geladen werden soll. Diese Einstellung wird später im Sceen-Projekt nochmals benötigt.

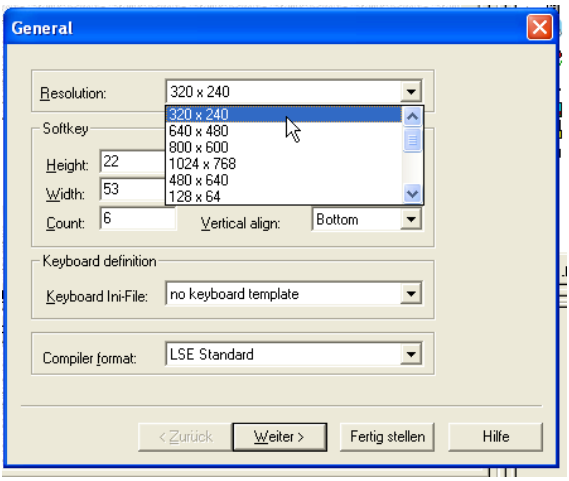


- Die restlichen Clients und Server werden fürs Erste nicht benötigt. Für eine Beschreibung der Funktion der Clients rufen Sie die Hilfe der Klasse auf.

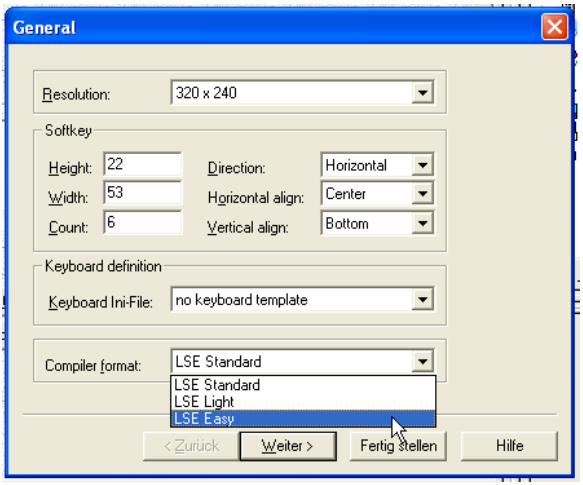
5 Erstellen eines Screen-Projekts zur Darstellung auf dem Display

- Erstellen Sie ein **neues Projekt.**

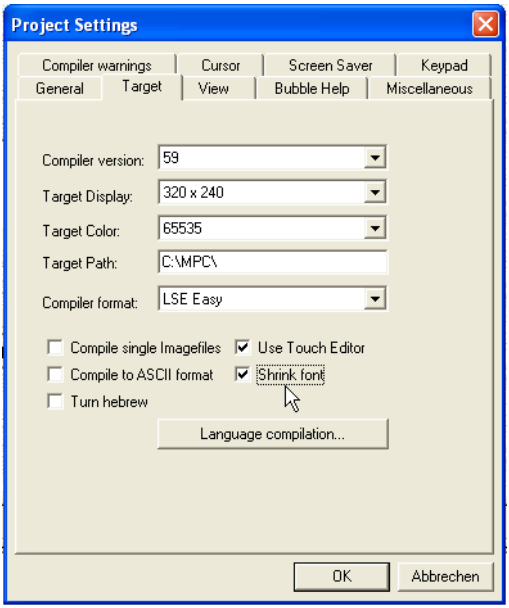
- Wählen sie bei der Resolution **320x240** aus.



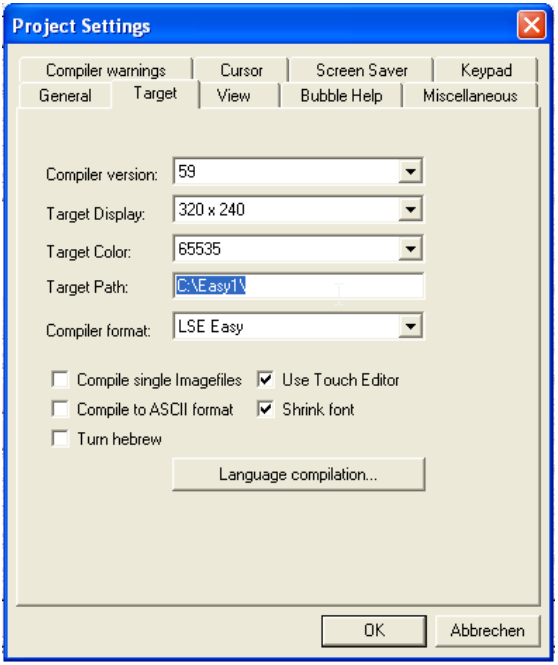
- Stellen Sie beim Compiler format: **LSE Easy** ein.



- Drücken Sie auf den Button **Fertig stellen.**
- Öffnen Sie die Project Settings.
- Wechseln Sie zu dem Reiter **Target.**
- Wählen Sie die Checkbox **Shrink font** an.



- Je nachdem, wie sie im Class Projekt den Client ProjectPath initialisiert haben, gehört nun der **Target Path** in den **Settings** verändert.



- Schließen Sie die Settings mit dem **OK** Button wieder.

- Fertig!

- Nun kann mit dem Erstellen der Visualisierung begonnen werden.

6 Bei der Verwendung des LSE Easy sind folgende Punkte zu beachten

- Es stehen nur Screens und der globale Screen zur Verfügung. Es können keine Windows oder Objekte verwendet werden.

- Bis Firmware 1.29 stehen dem User 20 Screens; ab FW 1.30 stehen dem User 40 Screens zur Verfügung.

- Daten-Server können nur von der eigenen CPU angezeigt werden (Reference To Variables\Connection Settings => intern).

- Units können verwendet werden, Umrechnung ab LSE Easy 1.19. Digits, Position of Decimal Point und Unit Text kann verwendet werden.

- Bitmaps so klein wie möglich verwenden. Es macht keinen Sinn, das Image mit 200x100 zu implementieren, wenn es dann mit 100x50 verwendet wird. Der Speicher ist knapp im Terminal. Maximal 256 Images/Files.

- Es stehen nur bestimmte Button-Frames zur Verfügung.

- Z-Order am Screen wird berücksichtigt.

- Sprachumschaltung ist möglich. Wird die Sprache umgestellt, wird die Visu von der PLC automatisch zum Display neu übertragen, da aus Speichergründen nicht alle Texte in allen Sprachen auf das Display übertragen werden. Anschließend startet das Display automatisch wieder mit der Ausgabe der Visualisierung.

- Mehrere Displays sind möglich. Es kann die gleich Visualisierung auf mehreren Displays dargestellt werden. Es kann auch für jedes Display eine eigene Visualisierung erstellt und dargestellt werden. Ein Mischbetrieb ist auch möglich (siehe Punkt 7).

- Beim Einschalten kann der SIGMATEK Setup Screen durch ein Bitmap ersetzt werden. Dazu muss im Screen-Projekt einfach ein Image mit dem Namen bootimage angelegt bzw. der in der Ableitung derLseEasy der Funktion GetBootimage() festgelegte Name verwendet werden. Um dann in das Setup zu kommen, drücken Sie während des boot images auf die rechte untere Ecke.

- Es steht eine Overload Funktion zur Verfügung um die gleiche Visualisierung für unterschiedliche Class Objekte anzuzeigen (siehe Punkt 8).

- String Editor steht ab Display FW 1.30 zur Verfügung.

- Real Zahlenwerte können NICHT korrekt dargestellt werden.

- Keine Color-Schemas (nur Unit-, Font-, Text- und Image-Schemas)

- Image Alignment für Buttons hardcoded auf CENTER/CENTER

7 Verwendung von mehreren Displays

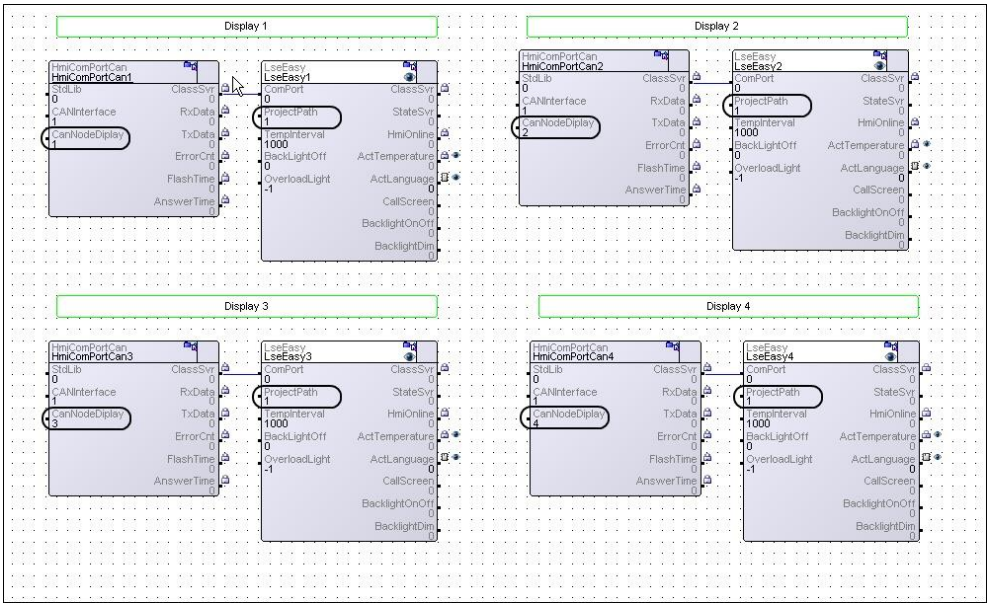
Jedes Display muss separat konfiguriert werden (CanNode, CanBaud). Für jedes Display muss ein eigenes Objekt der Klasse HmiComPortCan und LseEasy angelegt werden. Die CanNode-Einstellung der Displays muss auf den Client des entsprechenden Objektes eingestellt werden (CanNodeDisplay).

7.1 Mehrere Displays mit der gleichen Visualisierung

Es wird eine Visualisierung für das Display erstellt. In den Projekt-Einstellungen wird unter dem Punkt Target Path ein Pfad eingestellt.

Zum Beispiel: C:\Easy1\ => ProjectPath =1 für alle vier Displays

Dieser wird dann bei den Objekten der Klasse LseEasy konfiguriert. Der Client ProjectPath wird bei allen Objekten gleich eingestellt.



7.2 Mehrere Displays mit unterschiedlichen Visualisierungen

Es werden vier verschiedene Visualisierungen erstellt. Für jedes Display eine andere Visualisierung. In den Projekt-Einstellungen der einzelnen Visualisierungen werden unter dem Punkt Target Path unterschiedliche Pfade eingestellt.

Zum Beispiel:

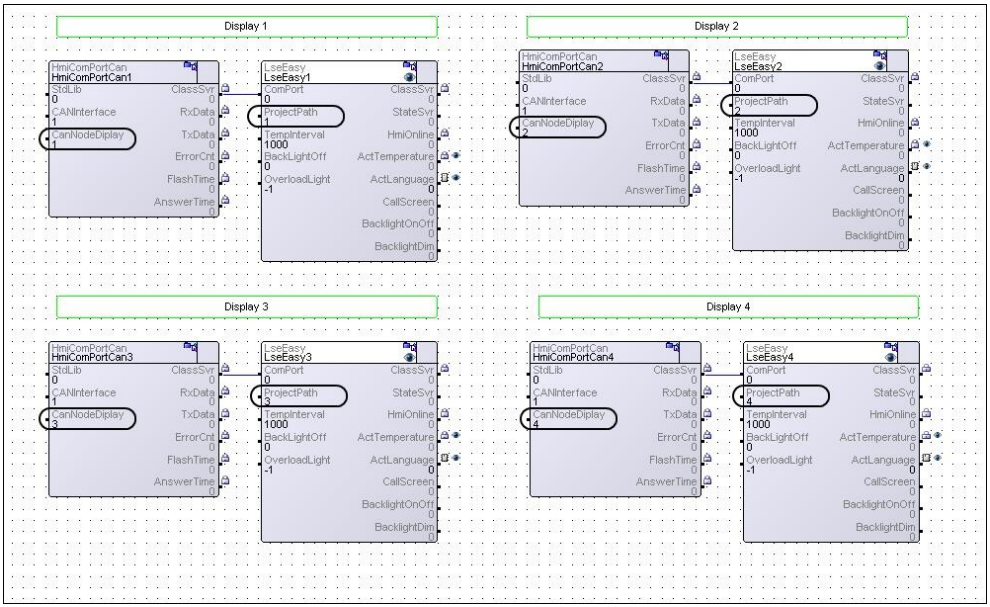
C:\Easy1\ => ProjectPath =1 für die Visualisierung 1 auf dem Display 1

C:\Easy2\ => ProjectPath =2 für die Visualisierung 2 auf dem Display 2

C:\Easy3\ => ProjectPath =3 für die Visualisierung 3 auf dem Display 3

C:\Easy4\ => ProjectPath =4 für die Visualisierung 4 auf dem Display 4

Diese werden nun bei den Objekten der Klasse LseEasy eingestellt. Der Client ProjectPath wird nun entsprechend der zugehörenden Visualisierung konfiguriert.



7.3 Mischbetrieb

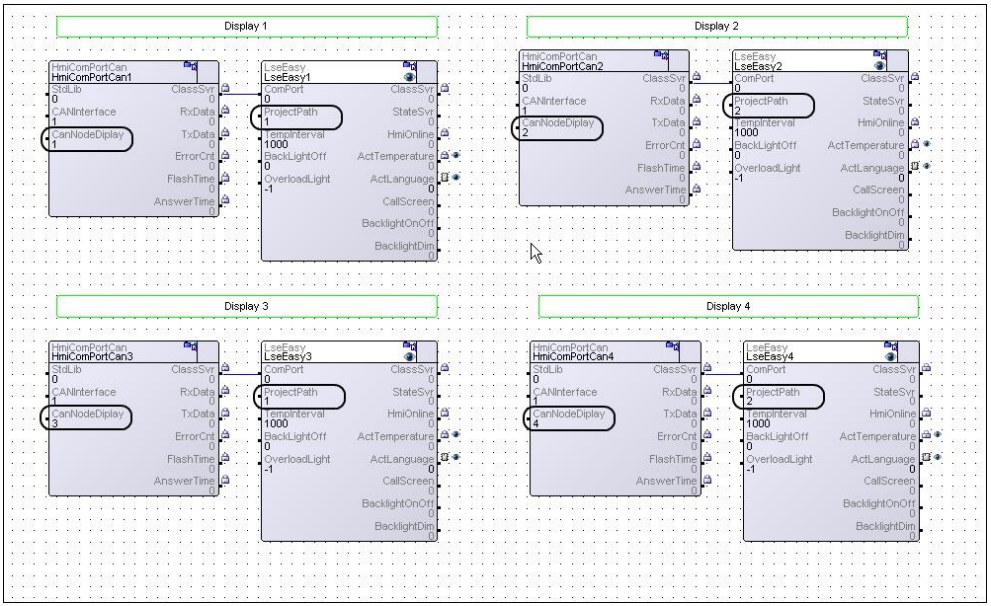
Ein Mischbetrieb ist ohne weiteres möglich. Es werden zwei verschiedene Visualisierungen erstellt. Für jeweils zwei Displays eine Visualisierung. In den Projekt-Einstellungen der zwei Visualisierungen werden unter dem Punkt Target Path unterschiedliche Pfade eingestellt.

Zum Beispiel:

C:\Easy1\ => ProjectPath =1 für die Visualisierung 1 auf dem Display 1 und 3

C:\Easy2\ => ProjectPath =2 für die Visualisierung 2 auf dem Display 2 und 4

Diese werden nun bei den Objekten der Klasse LseEasy eingestellt. Der Client ProjectPath wird nun entsprechend der zugehörenden Visualisierung konfiguriert.



8 Überladen von Daten für mehrere Displays mit einer Visualisierung

Jedes Display muss separat konfiguriert werden (CanNode, CanBaud).

Für jedes Display muss ein eigenes Objekt der Klasse HmiComPortCan und LseEasy angelegt werden. Die CanNode-Einstellung der Displays muss auf den Client des entsprechenden Objektes eingestellt werden (CanNodeDisplay).

Es wird eine Visualisierung für alle Displays erstellt. In den Projekt-Einstellungen wird unter dem Punkt Target Path ein Pfad eingestellt.

Zum Beispiel: C:\Easy1\ => ProjectPath =1 für alle vier Displays

Dieser wird dann bei den Objekten der Klasse LseEasy konfiguriert. Der Client ProjectPath wird bei allen Objekten gleich eingestellt.

Um auf jedem Display unterschiedliche Daten anzeigen zu können, wurde der Client OverloadLight der Klasse LseEasy eingeführt. Der Standardwert ist -1. Wertebereich ist 0 bis 99.

Wird hier eine Zahl eingestellt, so werden beim Laden des Projektes alle Objekt.Server in der Variablenliste durchsucht. Wird hier irgendwo die Zeichenfolge XX gefunden, so wird diese durch die Zahl, welche beim Client eingestellt ist, ersetzt. Wobei die Zahl immer auf 2 Stellen formatiert wird.

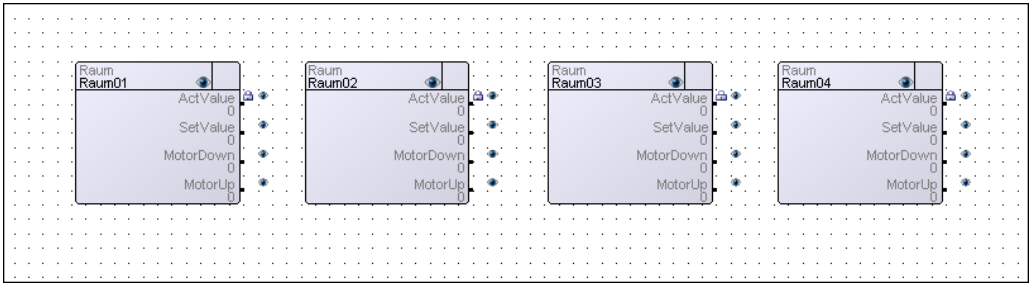
Beispiel: Client OverloadLight ist auf 9 Initialisiert.

ObjektXX.SetValue => wird auf Objekt09.SetValue umgewandelt.

Objekt.TempXXMax => wird auf Objekt.Temp09Max umgewandelt.

Hat man nun vier Displays, auf welchen die gleiche Visualisierung laufen soll, jedoch mit anderen Daten, so kann man dies wie folgt lösen:

Klasse im LASAL2 :	Raum	.ActValue	(Server)
		.SetValue	(Server)
		.MotorDown	(Server)
		.MotorUp	(Server)

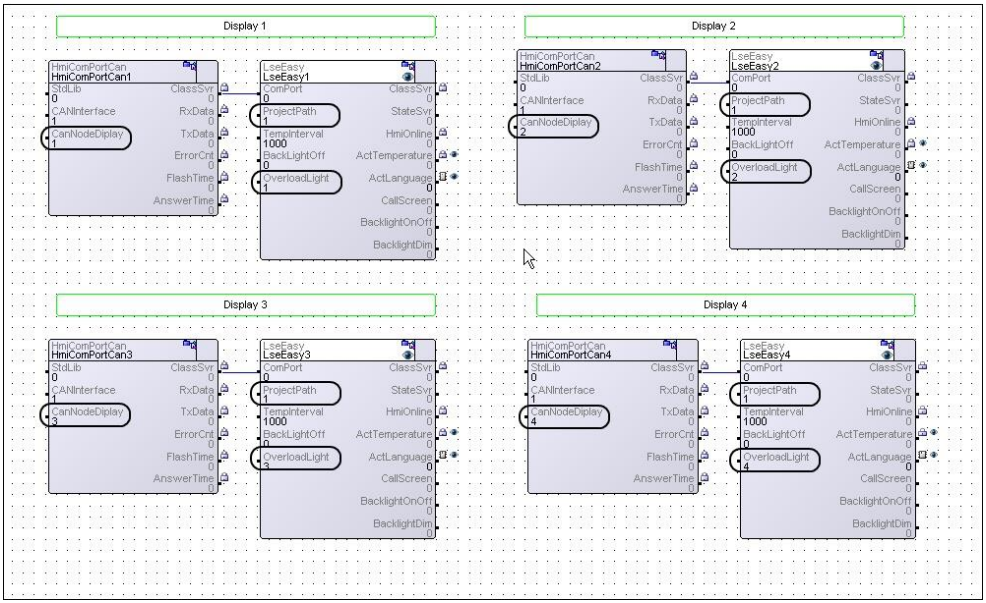


Objekte im LASAL2: Raum01, Raum02, Raum03, Raum04

Man importiert im Screen Editor die Objekte, löscht anschließend die Objekte Raum02, Raum03 und Raum04. Dann benennt man das Objekt Raum01 in RaumXX um.

Jetzt kann man die Server des Objektes RaumXX verwenden.

Im LASAL2-Projekt konfiguriert man den Client OverloadLight wie folgt:



9 CAN-Bus Protokoll

Grundsätzlich ist das Protokoll in zwei Schichten aufgeteilt.

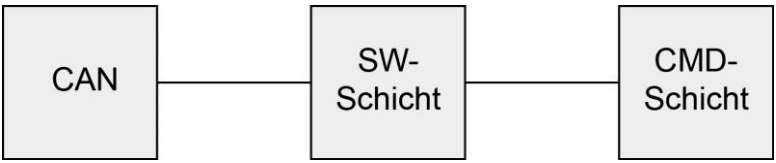
SW-Schicht

Bereitet die zu sendenden Daten (an HMI) für den CAN-Bus auf und gibt diese der CAN-Schnittstelle weiter.

Bereitet die empfangen Daten (vom HMI) vom CAN-Bus auf und gibt sie an die CMD-Schicht weiter (siehe Punkt 9.1).

CMD-Schicht

Bildet die ein einzelnen Kommandos ab (siehe Punkt 9.2).



9.1 SW-Schicht

In der CMD-Schicht wird das Kommando zusammengebaut. Diese Daten werden dann anhand einer SendData Methode an die SW-Schicht übergeben. Diese verschickt die Daten dann über den CAN-Bus. Die Antworten werden von dieser Schicht auch wieder empfangen und entsprechend ausgewertet, zusammengebaut und wieder an die CMD-Schicht übergeben.

Es werden folgende Objektnummern verwendet:

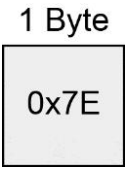
```
#define CAN_TX_OBJECT    0x020
#define CAN_RX_OBJECT    0x040
```

Zu diesen ID's muss noch die CAN-Node des Displays addiert werden.

Grundsätzlich gibt es 4 verschieden Paketarten:

- Acknowledge

#define ID_ACK_MESSAGE	0x7E	1 Byte



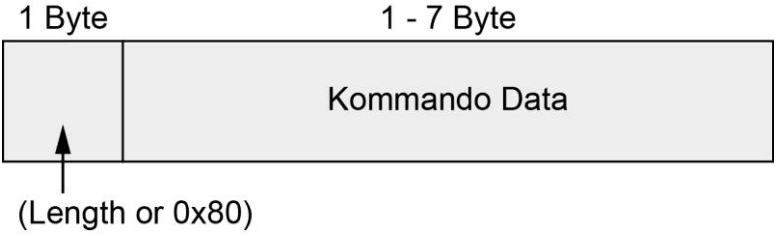
Beispiel:

```
unsigned char tmp;
tmp = ID_ACK_MESSAGE;
CanSend(tmp, 1);
```

- Single Pack Message

Ist dann der Fall, wenn der Kommandostream kleiner oder gleich 7 Byte groß ist. Jede Single Pack Message wird von der Gegenstelle mit einem Acknowledge beantwortet. Erst wenn das Acknowledge empfangen wurde, ist das Kommando sicher von der Gegenstelle angenommen worden.

Folgender Aufbau des Datenstreams ist notwendig:



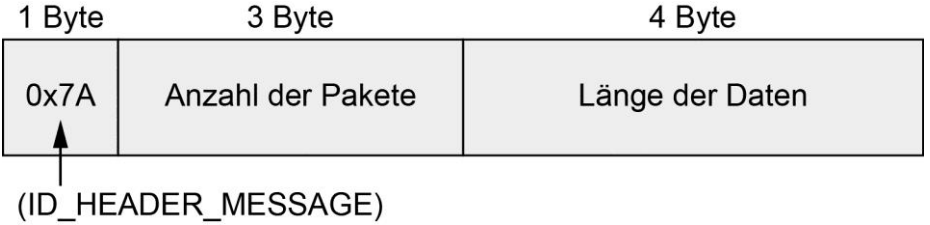
Beispiel:

```
if(mydatalengt <= 7)
{
 unsigned char tmp[8];
tmp[0] = mydatalength | 0x80; // userdatalänge + MSB=1
memcpy(&tmp[1], mydata, mydatalength);  // userdata
CanSend(tmp, 8); // senden
// WaitForAcknowledgeWithTimeout(); // auf Acknowledge von
Gegenstelle warten
}
```

- Multi Pack Message Header

Ist der Kommandostream größer als 7 Byte, so müssen die Daten in mehrere Pakete zerlegt werden. Das erste dieser Pakete muss dann wie folgt formatiert werden. Anschließend wird dieses Kommando von der Gegenstelle mit einem Acknowledge beantwortet. Erst wenn das Acknowledge empfangen wurde, ist das Kommando sicher von der Gegenstelle angenommen worden.

#define ID_HEADER_MESSAGE	0x7A	8 Byte



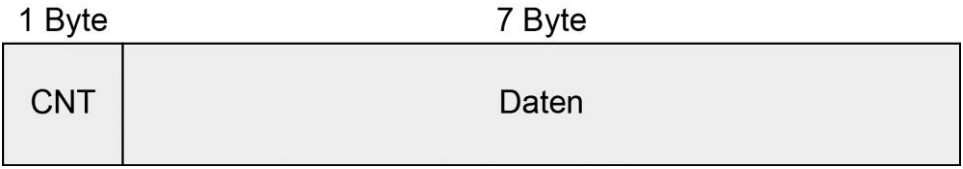
Beispiel:

```
if(mydatalength > 7)
{
unsigned char tmp[8];
tmp[0] = ID_HEADER_MESSAGE;
*(unsigned long*)&tmp[1] = (mydatalength + 6) / 7; // Anzahl
der folgenden Pakete
*(unsigned long*)&tmp[4] = mydatalength; // Gesamtlänge der
userdaten in Bytes
CanSend(tmp, 8);
// WaitForAcknowledgeWithTimeout(); // auf Acknowledge von
Gegenstelle warten
}
```

- Multi Pack Message

Es werden nun solange Pakete mit der Länge von 8 Byte (7 Byte Nutzdaten) verschickt, bis das letzte Paket an der Reihe ist. Dieses kann eine Länge von 1 – 7 Byte haben. Ist die Anzahl der Pakete größer als 8, so schickt die Gegenstelle nach jedem 8ten Paket ein Acknowledge zurück. Erst wenn dieses empfangen wurde, dürfen die nächsten Pakete geschickt werden (ansonsten müssen die letzten 8 Pakete wiederholt werden). Das letzte Paket muss nicht aus 7 Byte bestehen. Dies hängt davon ab, wie viele Byte noch übrig sind. Nach dem letztem Paket schickt die Gegenstelle ebenfalls ein Acknowledge zurück. Hat man auch dieses empfangen, so wurden die Daten erfolgreich übertragen.

Datenpaket (0..n-1)

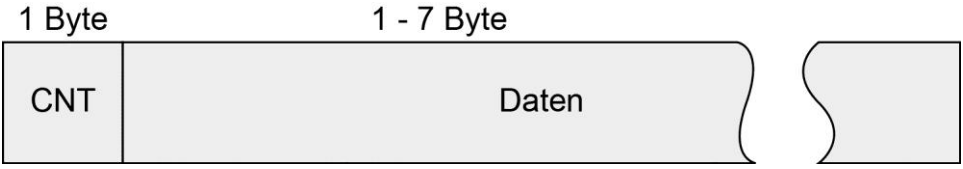


CNT => PaketIndex (0..7)

Beispiel:

```
unsigned char tmp[8];
tmp[0] = idx; // Paketindex (0-7)
memcpy(tmp[1], mydata, 7); //
CanSend(tmp, 8);
```

Datenpaket (n)



CNT => PaketIndex (0..7)

Beispiel:

```
unsigned char tmp[8];
tmp[0] = idx; // Paketindex (0-7)
memcpy(tmp[1], mydata, 3); //
CanSend(tmp, 4);
```

9.2 CMD-Schicht

Alle Kommandos und Datentypen für das HMI sind im File MiniSrcData.h für die programmtechnische Verwendung definiert.

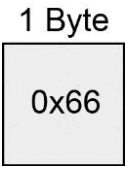
Anschließend sind die wichtigsten Kommandos beschrieben, alle anderen bedürfen spezieller Kenntnis und sollten nicht verwendet werden.

9.2.1 ComCMD_ALIVE PLC <=> HMI

Dieses Kommando ist ein sogenanntes Alive-Signal und signalisiert dem jeweiligen Empfänger die Präsenz (Funktionalität) des Senders. Dieses Kommando kann sowohl vom HMI an die PLC als auch umgekehrt gesendet werden. Die PLC sollte ca. alle 1000 ms Daten via CAN-Bus zum HMI senden. Sind aktuell keine Daten zum Senden an das HMI vorhanden, sollte dieses Kommando abgesetzt werden. Ansonsten erscheint im HMI eine Offline-Meldung, welche das Fehlen der PLC signalisiert.

Dieses Kommando benötigt keine zusätzlichen Parameter.

#define ComCMD_ALIVE	0x66	1 Byte



Beispiel:

```
unsigned char cmd =  ComCMD_ALIVE;
DataSend(&cmd, 1); // Call SW-Schicht
```

9.2.2 ComCMD_UPDATE PLC <=> HMI

Das Kommando Update tauscht Werte zwischen PLC und HMI aus.

Anhand einer vom LSE erstellten ID kann der Wert des entsprechenden Servers am Display verändert werden. Dieses Kommando wird ebenfalls bidirektional ausgeführt. Änderungen von Istwerten werden an das HMI gesendet und mittels Eingabe bestätigte Wertänderungen im HMI werden somit der PLC mitgeteilt.

Es ist zu beachten, dass alle mit diesem Kommando verschickten Werte als signed 32 Bit verarbeitet werden, d.h.: nur Werte von -2.147.483.648 bis +2.147.483.647 sind möglich.

#define ComCMD_UPDATE	0x10	7 Byte

1 Byte	2 Byte	4 Byte
0x10	ID	Value 32-Bit signed

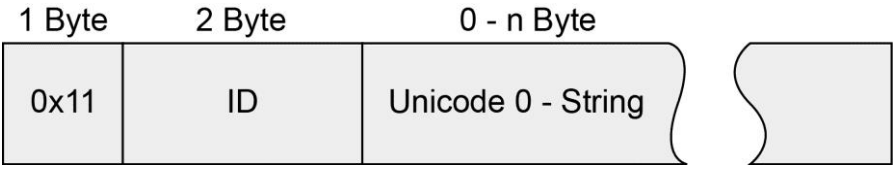
Beispiel:

```
unsigned char tmp[7];
tmp[0] = ComCMD_UPDAT;
*(unsigned short*)&tmp[1] = varid;
*(unsigned short*)&tmp[3] = value;
DataSend(tmp, 7); // Call SW-Schicht
```

9.2.3 ComCMD_UPDATESTRING PLC => HMI

Das Kommando UpdateString gibt es nur in eine Richtung, da Strings nur angezeigt, jedoch nicht eingegeben werden können. Anhand der ID erfolgt die Zuweisung zum entsprechendem Server. Es werden nur Unicode Strings berücksichtigt. ASCII String müssen entsprechend konvertiert werden.

#define ComCMD_UPDATESTRING	0x11	5 - n Byte



Beispiel:

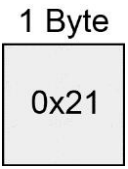
```
unsigned long textlen;
unsigned short uni_string [10];
uni_string[0] = 'H';
uni_string[1] = 'e';
uni_string[2] = 'l';
uni_string[3] = 'l';
uni_string[4] = 'o';
uni_string[5] = 0;

unsigned char tmp[255]
tmp[0] =  ComCMD_ALIVE;
*(unigned short*)&tmp[1] = id; // id of variable
textlen  =  (StrLenUni(uni_string)  +  1)  *  sizeof(unsigned
short);
memcpy(&tmp[3], uni_string, textlen);
DataSend(&tmp, textlen + 3); // Call SW-Schicht
```

9.2.4 ComCMD_RESET PLC => HMI

Wird verwendet um das Display in den Status RESET zu setzen.

#define ComCMD_RESET	0x21	1 Byte



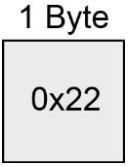
Beispiel:

```
unsigned char cmd =  ComCMD_RESET;
DataSend(&cmd, 1); // Call SW-Schicht
```

9.2.5 ComCMD_RUN PLC => HMI

Wird verwendet um das Display in den Status RUN zu setzen.

#define ComCMD_RUN	0x22	1 Byte



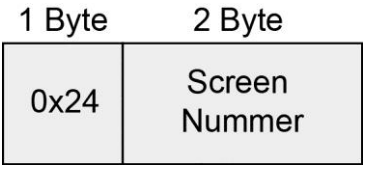
Beispiel:

```
unsigned char cmd =  ComCMD_RUN;
DataSend(&cmd, 1); // Call SW-Schicht
```

9.2.6 ComCMD_SCREEN PLC => RUN

Das Kommando wird verwendet um von der PLC aus die Bildschirmseite des Displays umzuschalten.

#define ComCMD_SCREEN	0x24	3 Byte



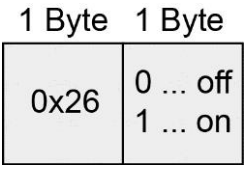
Beispiel:

```
unsigned char cmd[3] ;
cmd[0] = ComCMD_SCREEN;
*(unsignd short*)&cmd[1] = screennumber;
DataSend(&cmd, 3); // Call SW-Schicht
```

9.2.7 ComCMD_BACKLIGHT PLC => HMI

Damit kann die Hintergrundbeleuchtung des Display an- / ausgeschalten werden, wenn man nicht die eingebaute Funktion verwenden will (siehe ComCMD_BACKLIGHTTIME).

#define ComCMD_BACKLIGHT	0x26	2 Byte



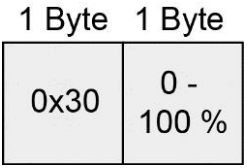
Beispiel:

```
unsigned char cmd[2] ;
cmd[0] = ComCMD_SCREEN;
cmd[1] = 1; // cmd[1] = 0;
DataSend(&cmd, 2); // Call SW-Schicht
```

9.2.8 ComCMD_BACKLIGHTDIM PLC => HMI

Mit diesem Kommando kann die Hintergrundbeleuchtung in der Helligkeit verstellt werden (Wert 0 ... 100 %, wobei 0 % nicht aus ist). Um das Display auszuschalten, verwenden Sie bitte das Kommando ComCMD_BACKLIGHT.

#define ComCMD_BACKLIGHTDIM	0x30	2 Byte



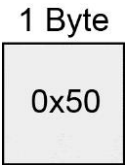
Beispiel:

```
unsigned char cmd[2] ;
cmd[0] = ComCMD_BACKLIGHTDIM;
cmd[1] = 70; // Backlight wird auf 70% gestellt.
DataSend(&cmd, 2); // Call SW-Schicht
```

9.2.9 ComCMD_ASK_TEMP PLC => HMI

Wird verwendet, um die aktuelle Temperatur des Displays anzufordern. Dieses antwortet dann mit dem Kommando ComCMD_TEMP.

#define ComCMD_ASK_TEMP	0x50	1 Byte



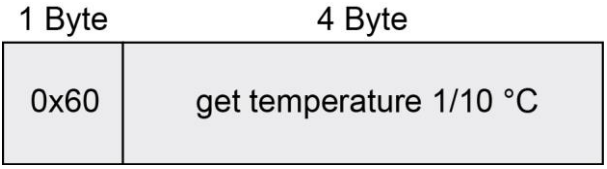
Beispiel:

```
unsigned char cmd ;
cmd = ComCMD_ASK_TEMP;
DataSend(&cmd, 1); // Call SW-Schicht
```

9.2.10 ComCMD_TEMP HMI => PLC

Dies ist die Antwort auf die Anfrage von der PLC (Kommando ComCMD_ASK_TEMP). Es wird die aktuelle Temperatur in Zehntel Grad Celsius geliefert.

#define ComCMD_TEMP	0x60	5 Byte



9.2.11 ComCMD_ASK_ALIVE PLC <=> HMI

Kann sowohl von der PLC als auch vom HMI gesendet werden. Als Antwort ist das Kommando ComCMD_ALIVE (siehe Punkt 9.2.1) zu schicken.

#define ComCMD_ASK_ALIVE	0x56	1 Byte

Beispiel:

```
unsigned char cmd ;
cmd = ComCMD_ASK_ALIVE;
DataSend(&cmd, 1); // Call SW-Schicht
```

9.2.12 ComCMD_ASK_ACTSCREEN PLC => HMI

Mit diesem Kommando ist es möglich die aktuell angezeigte Bildschirmseite auszulesen. Die Antwort kommt dann mit dem Kommando ComCMD_ACTSCREEN.

#define ComCMD_ASK_ACTSCREEN	0x59	1 Byte

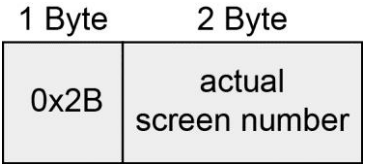
Beispiel:

```
unsigned char cmd ;
cmd = ComCMD_ASK_ACTSCREEN;
DataSend(&cmd, 1); // Call SW-Schicht
```

9.2.13 ComCMD_ACTSCREEN HMI => PLC

Dies ist die Antwort auf die Anfrage von der PLC (Kommando ComCMD_ASK_ACTSCREEN). Es liefert die aktuelle angezeigte Bildschirmnummer zurück.

#define ComCMD_ACTSCREEN	0x2B	3 Byte

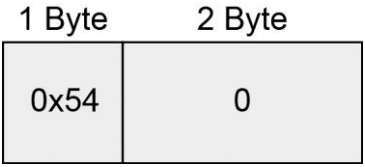


9.2.14 ComCMD_ASK_FILE_CRC PLC => HMI

Wird benötigt, um die aktuelle CRC des am Display gespeicherten Projektes abzurufen. Dies sollte überprüft werden, bevor mit dem Update der Variablen gestartet wird, da, wenn die CRC nicht identisch mit jener ist, welche im File EaysMap.txt enthalten ist, die ID eventuell nicht gleich ist. Die Antwort kommt als Kommando ComCMD_FILE_CRC.

Nach jedem Kompiliervorgang des LSE können sich theoretisch die IDs der Server ändern. Deshalb sollte die ID nicht fix im Code codiert werden, sondern als Konstante angelegt werden. Um die Projekt CRC zu erhalten, ist es notwendig bei den Daten 0 zu übergeben.

#define ComCMD_ASK_FILE_CRC	0x54	3 Byte



Beispiel:

```
unsigned char cmd[2] ;
cmd[0] = ComCMD_ASK_FILE_CRC;
cmd[1] = 0; // um die Projekt CRC zu bekomen.
DataSend(&cmd, 2); // Call SW-Schicht
```

9.2.15 ComCMD_FILE_CRC HMI => PLC

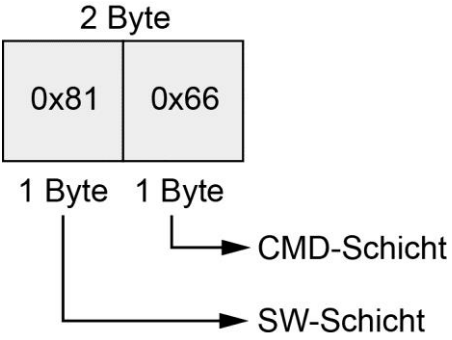
Dies ist die Antwort auf das Kommando ComCMD_ASK_FILE_CRC und liefert die CRC des aktuell gespeicherten Projekts auf dem Display zurück.

#define ComCMD_FILE_CRC	0x64	5 Byte

10 Kommandobeispiele

- Es soll das ComCMD_ALIVE Kommando abgesetzt werden.

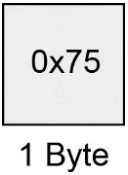
Die Daten, welche auf den CAN-Bus geschickt werden müssen, sehen wie folgt aus:




```
0x66 => Kommando  ComCMD_ALIVE (1Byte Länge)
0x81 => Single Pack Message (0x80 OR 0x01)
       0x80 => Kennung Single Pack Message
       0x01 => Länge des  ComCMD_ALIVE)
```

Das bedeutet, diese 2 Byte müssen mit der Objektnummer 0x20 + CanNode Display verschickt werden.

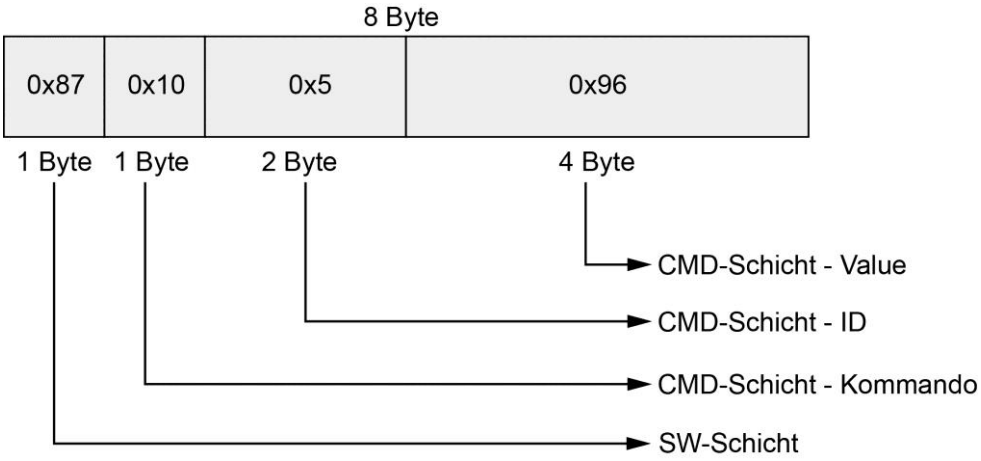
Die Gegenstelle antwortet mit einem Acknowledge. Dieses wird auf der Objektnummer 0x40 + CanNode Display empfangen und sieht wie folgt aus:



Es wird 1 Byte empfangen. Der Inhalt der Daten ist 0x75. Dies entspricht der Paketart Acknowledge. ID_ACK_MESSAGE => 0x75

- Es soll ein Wert auf dem Display aktualisiert werden. Dazu wird das Kommando ComCMD_UPDATE benötigt. Es soll die Variable mit der ID : 5 verändert werden.

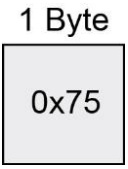
  Der Wert der Variable soll auf 150 geändert werden.




```
0x96 => Wert (150)
0x05 => ID des Servers ( 5)
0x10 => Kommando  ComCMD_UPDATE (7 Byte )
0x87 => Single Pack Message ( 0x80 OR 0x07 )
       0x80 => Kennung Single Pack Message
       0x07 => Länge des   ComCMD_UPDATE
```

Das bedeutet, dass diese 8 Byte mit der Objektnummer 0x20 + CanNode Display verschickt werden müssen.

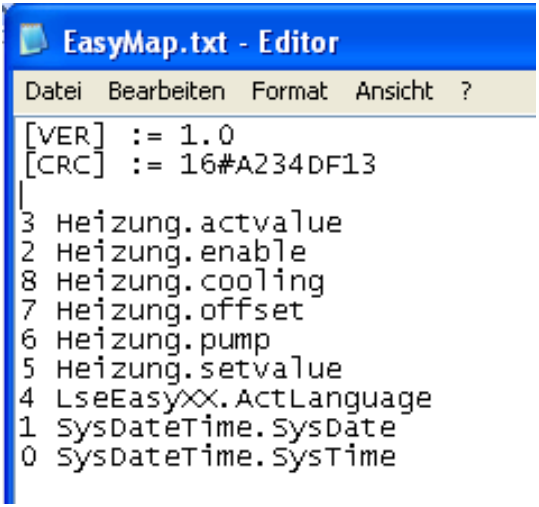
Die Gegenstelle antwortet mit einem Acknowledge. Dieses wird auf der Objektnummer 0x40 + CanNode Display empfangen und sieht wie folgt aus.



Es wird 1 Byte empfangen. Der Inhalt der Daten ist 0x75. Dies entspricht der Paketart Acknowledge. ID_ACK_MESSAGE => 0x75

11 EasyMap.txt

Dieses File stellt die Referenz zu den in der Visualisierung verwendeten Server her (File Format: regular expression). Dieses File wird von der Klasse LseEasy erstellt. Es befindet sich in dem Pfad auf der PLC, welcher beim Client ProjectPath eingestellt ist.



[CRC] => Checksummer des Projekts

3 .. ID des Servers : Heizung.actvalue

Änderungen der Dokumentation

Änderungs-datum	Betroffene Seite(n)	Kapitel	Vermerk
28.04.2016		6 Bei der Verwendung …	FW Updates